Linked Server 이렇게 사용하자!

㈜엑셈 컨설팅본부/ SQL Server팀 김 성식

Q(고객) : 연결된 서버를 사용하는 쿼리인데 실행시간이 굉장히 오래 걸리네요.
1 분 이상 걸립니다. 왜 그런가요????
(아.. MSSQL Server 못 쓰겠네!!!)

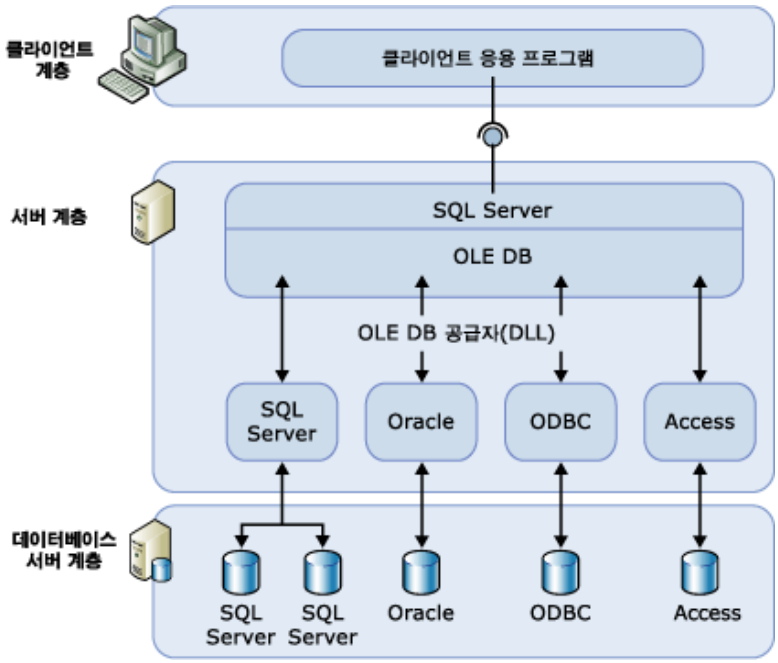
A(나) : 흠... 왜 그럴까요? ^^ㅋㅋ
연결된 서버가 무엇인지 간단히 알아보고, 정말 문제점이 MSSQL Server 이기 때문인가?
개선방법은 존재하지 않는 것인가? 차근차근 알아보자.

Linked Server의 용도

SQL Server 데이터베이스 엔진에서 SQL Server 인스턴스 외부의 OLE DB 데이터 원본에 대해 명령을 실행할 수 있도록 연결된 서버를 구성한다. 일반적으로 연결된 서버는 데이터베이스 엔진에서 SQL Server 의 다른 인스턴스 또는 Oracle 과 같은 다른 데이터베이스 제품에 있는 테이블이 포함된 Transact-SQL 문을 실행할 수 있도록 구성된다. Microsoft Access 및 Excel 을 포함한 많은 유형의 OLE DB 데이터 원본을 연결된 서버로 구성할 수 있으며 연결된 서버에는 다음과 같은 이점이 있다.

- SQL Server 외부에서 데이터에 액세스할 수 있음.
- 기업 전체에 걸쳐 유형이 다른 데이터 원본에 대해 분산 쿼리, 업데이트, 명령, 트랜잭션 등을 수행할 수 있음.
- 다양한 데이터 원본을 유사하게 처리할 수 있음.

다음 그림은 연결된 서버 구성의 기본 사항을 보여 준다.



연결된 서버는 일반적으로 분산 쿼리를 처리하는 데 사용된다.

여기서 분선 쿼리란 다른 여러 데이터 원본의 데이터를 액세스하는 것을 의미하며 이러한 데이터 원본은 동일 컴퓨터나 다른 컴퓨터에 저장될 수 있다.

클라이언트 응용 프로그램이 연결된 서버를 통해 분산 쿼리를 실행할 때 SQL Server 는 명령을 구문 분석하고 OLE DB 로 요청을 보낸다. 행 집합 요청은 공급자에 대해 쿼리를 실행하거나 공급자로부터 기본 테이블을 여는 형식일 수 있다.

연결된 서버를 통해 데이터를 반환하는 데이터 원본의 경우 해당 데이터 원본에 대한 OLE DB Provider(DLL)는 SQL Server 인스턴스와 같은 서버에 있어야 한다.

타사 OLE DB Provider 를 사용하는 경우 SQL Server 서비스가 실행되는 계정에는 공급자가 설치된 디렉터리 및 모든 하위 디렉터리에 대한 읽기 및 실행 권한이 있어야 한다.

Linked Server가 미치는 성능이슈.

기술지원을 수행하다 보면 고객 담당자에게 이러한 질문을 많이 받는다.

"저희가 Oracle 를 Linked Server 로 구성하여 사용 중인데 성능이 느려요."

원인을 분석하면 크게 3 가지 유형의 패턴을 찾을 수 있었다.

첫째, Linked Server(Oracle)을 사용한 최적화되지 못한 쿼리.

사용되는 쿼리는 Oracle 로 요청되는 쿼리로써 Oracle 환경에 최적화가 되어져야 한다는 사실을 이해하지 못하는 경우.

둘째, 네트워크 성능 이슈.

셋째, 비효율적인 옵티마이저의 실행계획.

세 번째 부분이 이번 백서에서 중점으로 다룰 소재이며,

실제 사례를 통해 살펴보도록 하자.

```
문제 쿼리>
SELECT A.ZUONR AS OrderNo
        이하 생략~
FROM HANDW.HANDW.HANDWADM.BSID AS A WITH (NOLOCK)
LEFT JOIN HANDW.HANDW.HANDWADM.VBRP AS B WITH (NOLOCK) ON A.BELNR = B.VBELN AND
B.POSNR % 100 = 0
WHERE A.BUDAT BETWEEN @sDate + ' 00:00:00.000' AND @eDate + ' 23:59:59.999'
        AND A.KUNNR IN (SELECT RetailStoreCd FROM @SaleChnl_T)
        AND A.BLART IN ('RV', 'DR')
```

Q(고객) : 연결된 서버를 사용하는 쿼리인데 실행시간이 굉장히 오래 걸리네요.

1 분 이상 걸립니다. 왜 그런가요???? (아.. MSSQL Server 못 쓰겠네!!!)

A(나) : 흠… 왜 그럴까요? ^^ㅋㅋ 한번 상태를 확인해볼가요?

그로부터 잠시 후….

A(나) : 원격 테이블을 가져오는 단계에서 부하가 발생되고 있네요. 조인 후행 테이블을 원격서버로부터 많은 건수의 데이터를 가져오는 단계에서 성능저하가 발생하고 있네요.

Q(고객) : 그럼 어떻게 하면 되지요?

A(나) : 연결된 서버를 사용하여 원격서버의 결과셋을 가져올 때는 Cursorfetch 동작을 하게됨으로 네트웍 I/O 가 과다하게 발생되어 성능이 저하 됩니다. 연결된 서버를 반대로 구성하여 값이 적은 서브 쿼리 테이블((**@SaleChnl_T)** 값을 가져다가 HANDW 에서 수행되 도록 하면 두 테이블 간에 LEFT OUTER JOIN 한 결과 셋만 전송. 엄청난 네트웍 I/O 를 개선 할 수 있겠어요.

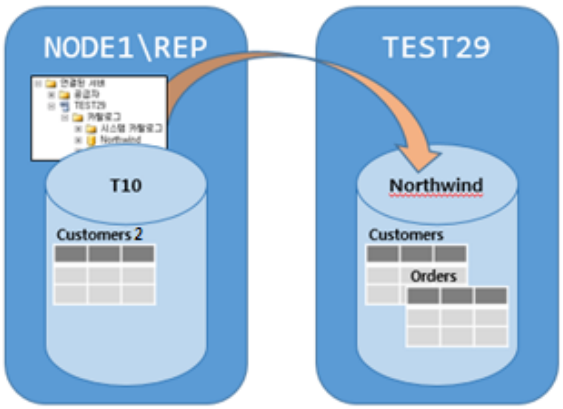
Q(고객) : 아하 그렇군요. 적용하니 잘 됩니다.
감사합니다.

문제 쿼리 플랜의 일부이다.

```
2131234        1    |--Remote Query(SOURCE:(HANDW), QUERY:(SELECT
"Tbl1003"."VBELN" "Col1115","Tbl1003"."FKIMG"
"Col1117","Tbl1003"."MATNR" ,"Col1118","Tbl1003"."ARKTX"
"Col1119","Tbl1003"."KZWI1" "Col1120" FROM "HANDW"."HANDWADM"."VBRP"
"Tbl1003" WITH (NOLOCK) WHERE CONVERT(int,"Tbl1003"."POSNR",0)%(100)=(0) ORDER BY
"Col1115" ASC))
```

HANDW.HANDWADM.VBRP 테이블 전체가 조회되는 상황에서 Cursorfetch 동작원리로 인한 과도한 네트웍 I/O 가 성능저하 문제의 원인이다.

이해를 좀더 돕기 위해, 간단한 테스트로 동작방식을 확인해 보자. 테스트 환경은 아래와 같으며, NODE1\REP 서버에서 TEST29 연결된 서버를 이용하여 쿼리가 수행된다.



```
select COUNT(*) from test29.Northwind.dbo.customers              -- 91 건
select COUNT(*) from test29.Northwind.dbo.orders                 -- 186,734 건

DBCC freeproccache

#테스트 쿼리>
select * from test29.Northwind.dbo.customers a
left outer join test29.Northwind.dbo.orders b
on a.customerid = b.customerid
where a.phone in (select phone from customers2 where customerid <> 'RICAR' )
```

TEST29 서버 customers 테이블 총 91 건, orders 테이블 186,734 건.
NODE1\REP 서버의 customers2 테이블과 TEST29 서버의 customers 테이블 정보는 일치하며 'RICAR' 고객을 제외한 고객들의 주문정보를 확인하는 쿼리이다.

```
#실제 실행 계획 : 요약>
6512   1          |--Merge Join(Left Outer Join,
90     1               |--Sort(ORDER
BY:([test29].[Northwind].[dbo].[customers].[CustomerID] ASC))
90     1               |    |--Merge Join(Inner Join,
90     1               |         |--Sort(DISTINCT ORDER
BY:([T10].[dbo].[customers2].[Phone] ASC))
90     1              |         |    |--Table Scan(OBJECT:([T10].[dbo].[customers2]),
0      0               |         |--Compute
91     1               |              |--Remote Query(SOURCE:(test29), QUERY:(SELECT
0      0               |--Compute
186734 1                    |--Remote Query(SOURCE:(test29), QUERY:(SELECT
"Tbl1003"."OrderID"

#IO or 수행시간 >
(6512 개행이영향을받음)
테이블'customers'. 검색수 1, 논리적읽기수 4, 물리적읽기수 0, 미리읽기수 0, LOB 논리적읽기수 0,
LOB 물리적읽기수 0, LOB 미리읽기수 0.

(10 개행이영향을받음)
(1 개행이영향을받음)
SQL Server 실행시간:
CPU 시간= 921 밀리초, 경과시간= 20083 밀리초
```

TEST29 서버의 customers 테이블과 NODE1\REP 서버의 customers2 테이블와의 조건에서 얻어진
90 건과 TEST29 서버의 orders 테이블 186,734(Table Scan)건을 Merge Join 하여 최종 6512 건을 조회하였다.

실제 사례와 동일하게 조인 후행테이블이 Table Scan 되어 많은 네트웍 I/O 및 과다 실행시간을 유발하였다.

어떻게 하면 좋을까?
1. 연결된 서버는 원격서버와 통신을 함으로 네트워크가 불안정한 것 같은데요?
2. 과도한 연결된 서버 사용은 Microsoft 가 권장하지 않습니다.
흠.. 그럴싸하다. ^^;;; 하지만 아무런 근거자료 없이 이러한 답변을 하면 서두의 대화글처럼 못 쓰겠다라는 말이 나올 것이다. 우리가 아무리 비싸고 좋다라는 DSLR 카메라를 샀지만 사용법을 익히지 못한다면 똑딱이 보다 못한 물건이 되는 것과 동일하다.

우리는 그래도 배운 사람들이고(ㅋ) 원인이 무엇인지 알았으니 좋은 방법이 없는지 생각해보자.

Linked Server의 성능이슈 개선방법.

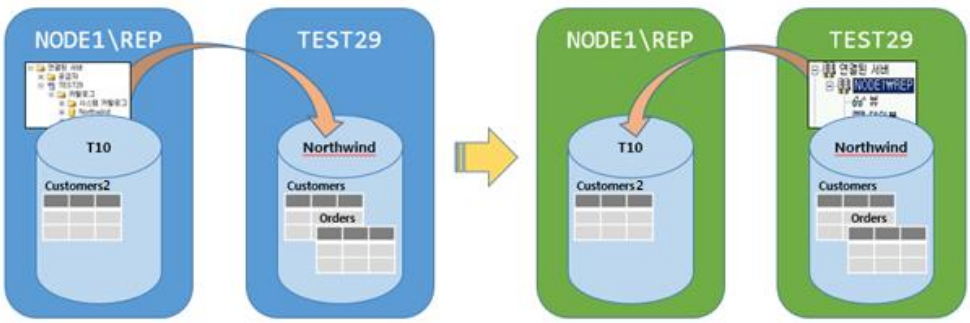
문제 쿼리의 패턴을 분석해 보면 조회대상 테이블들이 한 서버에 존재하는 것이 아니라 서로 다른 원격서버에 존재함으로 TEST29 서버의 customers 테이블과 NODE1\REP 서버의 customers2 테이블에서 얻은 결과로 TEST29 서버의 orders 테이블과 조인을 하게 되는데, 여기서 orders 테이블을 가져오는 단계에서 Cursorfetch 동작에 의한 과도한 네트웍 I/O 가 발생하게 된다.

EventClass	Reads	Writes	TextData
SQL:StmtCompleted	2	0	SELECT "Tbl1003","OrderID" "Col1061","Tbl1003","CustomerID" "Col1062","Tbl
Execution Plan			Execution Tree ---------- Sort(ORDER BY:([Tbl1003].[CustomerID]
RPC:Completed	570462	0	declare @P1 int set @P1=5 declare @P2 int set @P2=180150001 declare @P
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 1, 1
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 2, 100
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 102, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 202, 100
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 302, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 402, 100
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 502, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 602, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 702, 100
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 802, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 902, 100
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 1002, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 1102, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 1202, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 1302, 100
RPC:Completed	2	0	exec sp_cursorfetch 180150001, 16, 1402, 100
RPC:Completed	3	0	exec sp_cursorfetch 180150001, 16, 1502, 100

```
Execution Tree
--------------
Sort(ORDER BY:([Tbl1003].[CustomerID] ASC))
  |--Table Scan(OBJECT:([Northwind].[dbo].[Orders] AS [Tbl1003]))
```

결국, 네트웍 I/O 를 최소화 할 수 있는 솔루션을 찾으면 문제를 해결할 수 있을 것이다.

해서 아래와 같이 NODE1\REP 서버에서 프로시져를 요청하는 방식으로 NODE1\REP 서버에 있던 연결된 서버를 제거하고, TEST29 서버에다 NODE1\REP 서버로의 연결된 서버를 구성하여 TEST29 서버에서 NODE1\REP 서버 customers2 테이블의 91 건을 Remote Query 후에 조인되도록 유도하였다. 그 결과, 결과 값 6512 건만 NODE1\REP 서버로 리턴하게 되었으며 서버간에 네트웍 I/O 감소와 네트웍 환경에 민감한 연결된 서버 사용은 효율적이게 되었다.




```
# 테스트 >
NODE1\REP 서버에 아래 프로시져를 생성하고 TEST29 서버에서는 프로시져를 실행.

-- TEST29 : 저장 프로시져 생성
create  procedure Lk
as
```

```
select * into #tmp from (select phone  from [node1\rep].T10.dbo.customers
where customerid <> 'RICAR') a
select * from customers  a left outer join orders b
on a.customerid = b.customerid
where a.phone in (select phone from #tmp);

-- NODE1\REP : 저장 프로시져 실행
execute test29.Northwind..Lk

실제 실행 계획 >
6512    1         select * from {IRowset 0x20E2A58100000000}
6512    1           |--Remote Scan(OBJECT:(STREAM))

(6512 개행이영향을받음)
(2 개행이영향을받음)
(1 개행이영향을받음)
SQL Server 실행시간:
CPU 시간= 47 밀리초, 경과시간= 512 밀리초
SQL Server 실행시간:
CPU 시간= 47 밀리초, 경과시간= 518 밀리초
```

그리고 NODE1\REP 서버에서 TEST29 서버로의 프로시져(그림_1)와 TEST29 서버에서 프로시져 내부 조회쿼리(그림_2)를 각각 실행시켜 실행계획을 비교하니 동일한 플랜을 보였다. 즉, Cursorfetch 동작방식으로 인한 과도한 네트웍 I/O 가 원인임을 뒷받침해주는 증거자료가 되겠다.

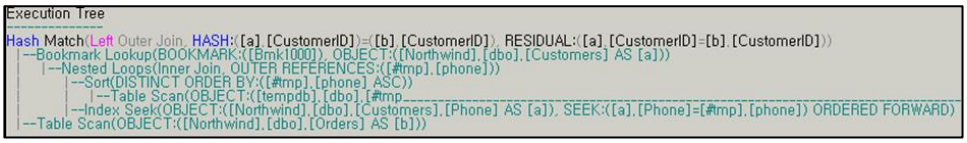


그림_1

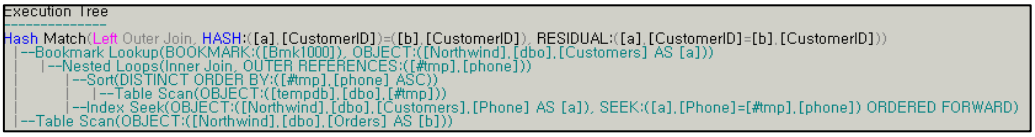


그림_2

마치며

실행 시간이 약 40 배 가량 개선된 것을 확인하였다.

이렇듯 DB 튜닝은 크게 어려운 것이 아니다. 우리가 일상생활에서 PC 와 자동차 등을 튜닝하듯이 DB 튜닝 또한 약간의 기본 전문지식 + 관심과 이해가 전부라 생각한다.

아무리 뛰어난 제품에도 장점만이 있는 것은 아니다. 단점 또한 존재하지만 그것을 대체 가능한 솔루션이 존재한다면 단점이라 말하기 힘들다. 고정관념을 깨는 것 또한 튜닝의 필수요소! 앞으로 우리는 옵티마이져의 동작 방식을 이해하고 상황에 맞게 적절한 대체 솔루션찾는 태도로 임하자.